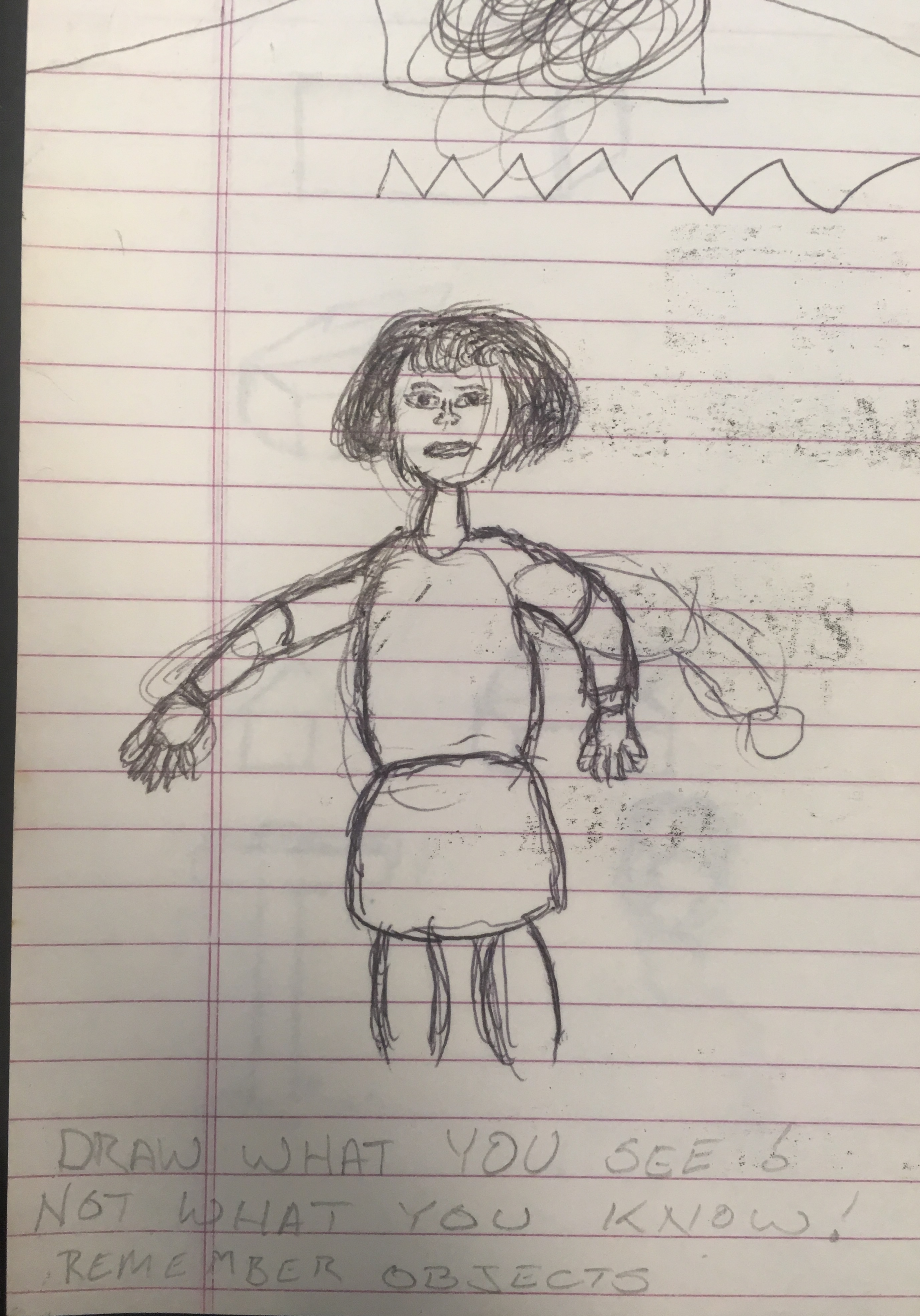
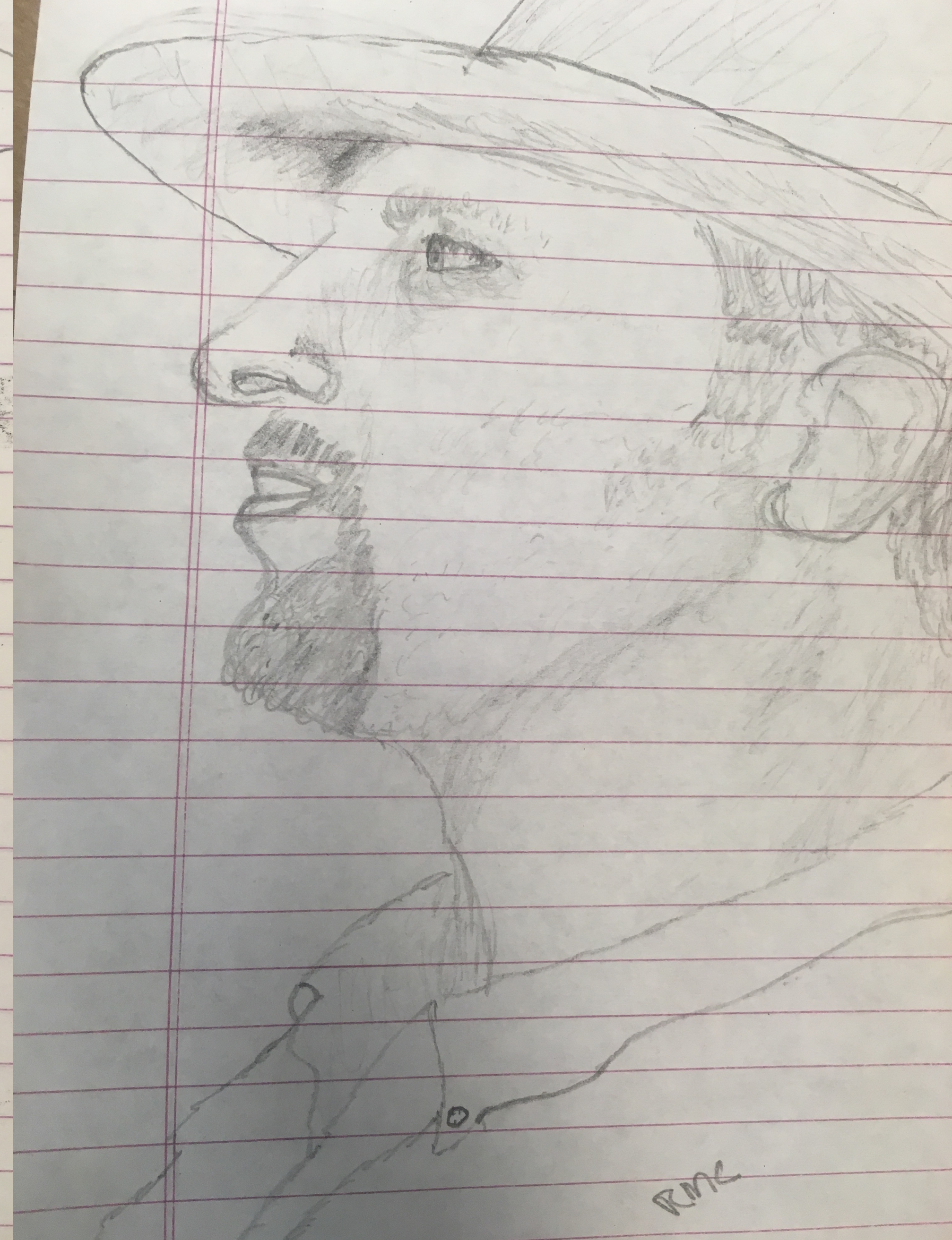
Finding structure in large datasets of particle distribution functions using unsupervised machine learning

R.M. Churchill
C.S. Chang, S. Ku


DRAW WHAT YOU SEE &
NOT WHAT YOU KNOW!
REMEMBER OBJECTS


RMC

Unsupervised Machine Learning

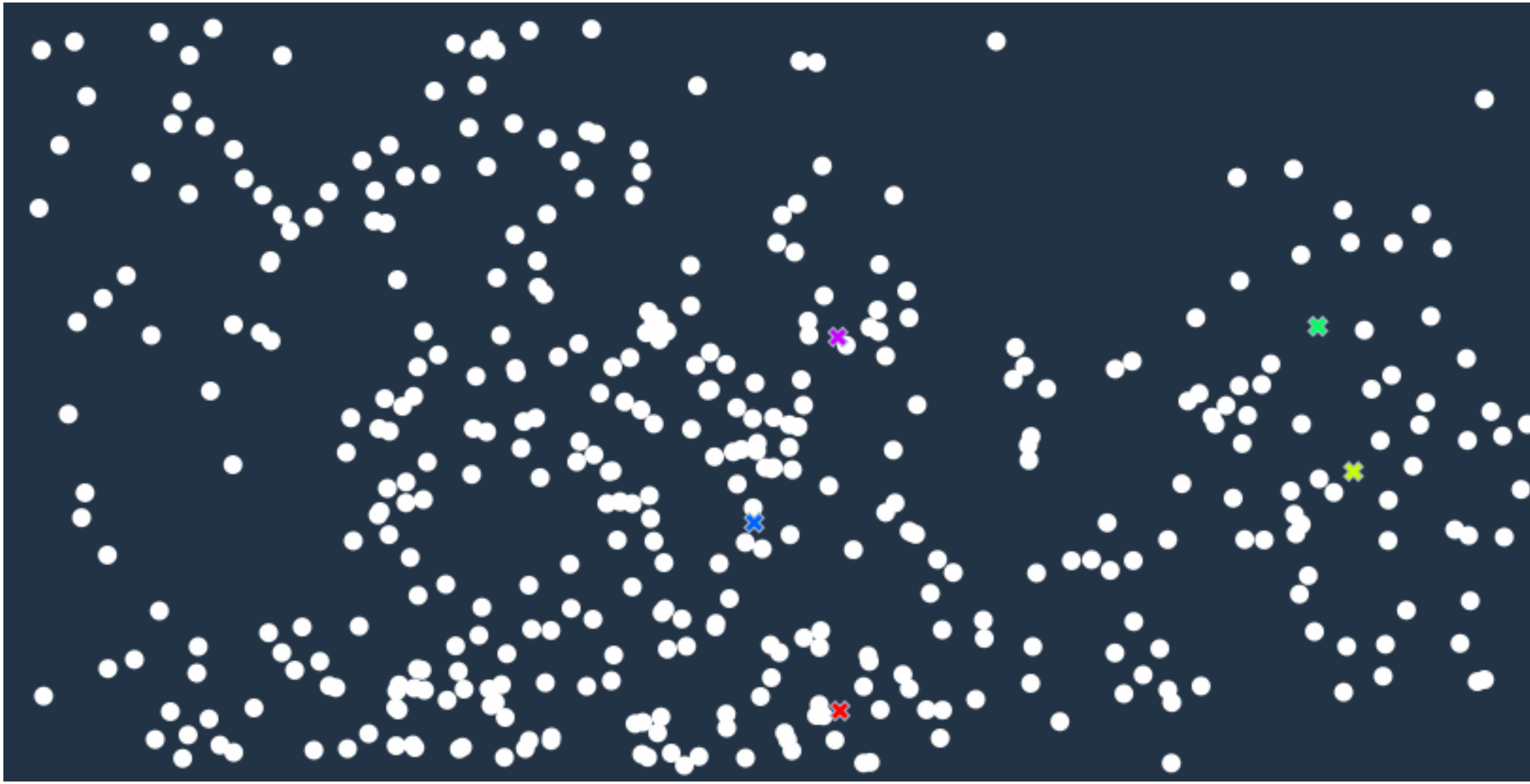
- Allows finding hidden structure in large data sets with little or no apriori knowledge

- A lot of focus on "supervised" machine learning, i.e. learning using labeled data

  unsupervised learning "next frontier" [LeCunn 2016]

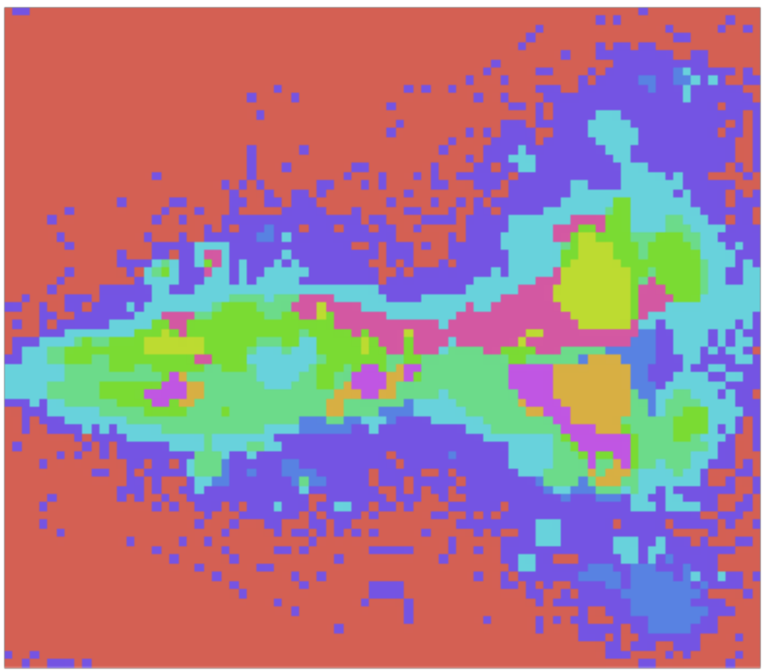
- Examples include:
  - Clustering (K-means, Gaussian Mixture Models, hierarchical,)
  - Dimensionality reduction (PCA, ICA, T-SNE)
  - Neural networks (autoencoders, adversarial networks)

K-means clustering

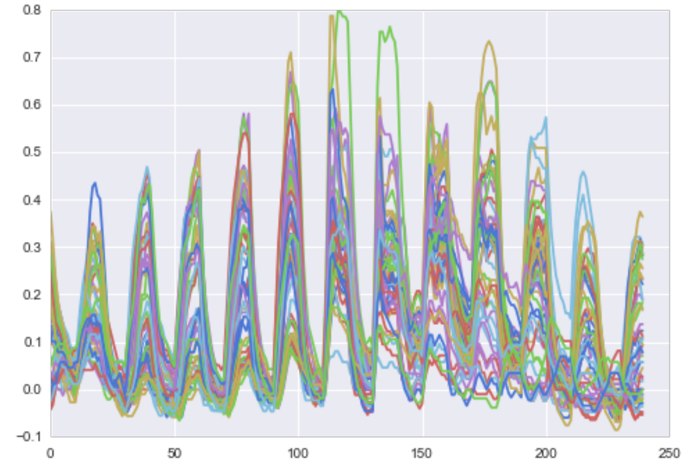


Example series k-means clustering from neuroscience

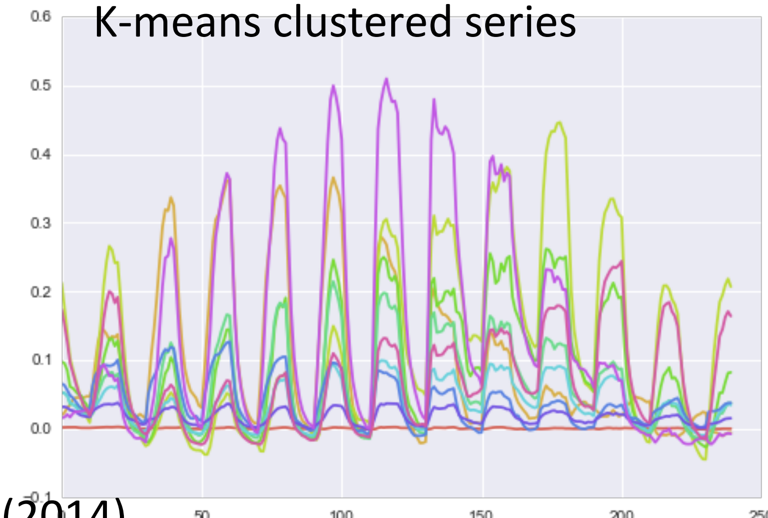
Detect neurons with time-series which have high correlations



Original data series



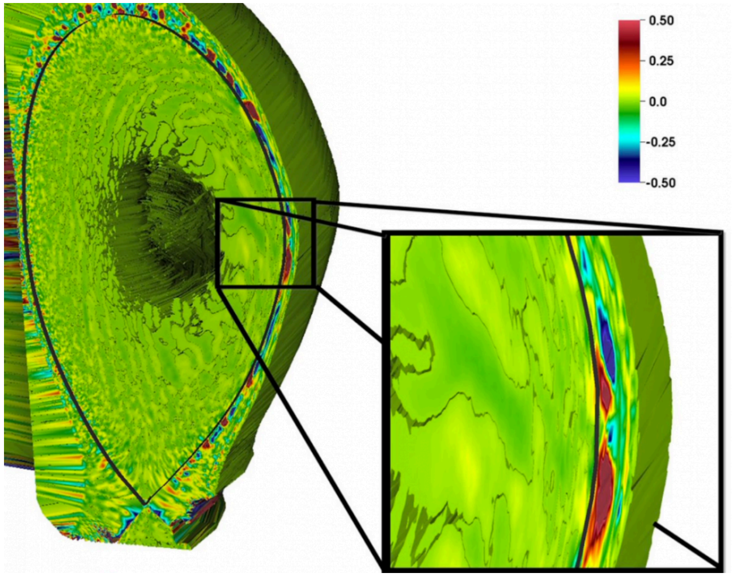
K-means clustered series



Freeman, Nature Methods 11, 941–950 (2014)

XGC1

- Full-f, gyrokinetic turbulence code focused on the edge (pedestal + SOL):
  - Neutrals, collisions, sheath physics, etc.
- Massively parallel, requires 100M+ CPU hours (HPC)
- Generates TB's of data per simulation



How to extract useful information?
Natural candidate for unsupervised machine learning

Apache Spark + Thunder: Image and time series distributed computing streamlined

PROS

- Distributed computing, easily scale up analysis
- Simple interface, Python bindings
- Resiliency
- Available on NERSC
- Machine learning libraries (MLlib) optimal for parallel processing

CONS

- Networking slower than MPI
- Complex communication patterns are difficult to implement (better for embarrassingly parallel)
- Learning curve

Spark code, reading of scientific data

- Read data in batches from parallel file system, split in-place for individual records

- Single node (22 cores) gave data reading scaling of 1 GB/s up to 33 GB

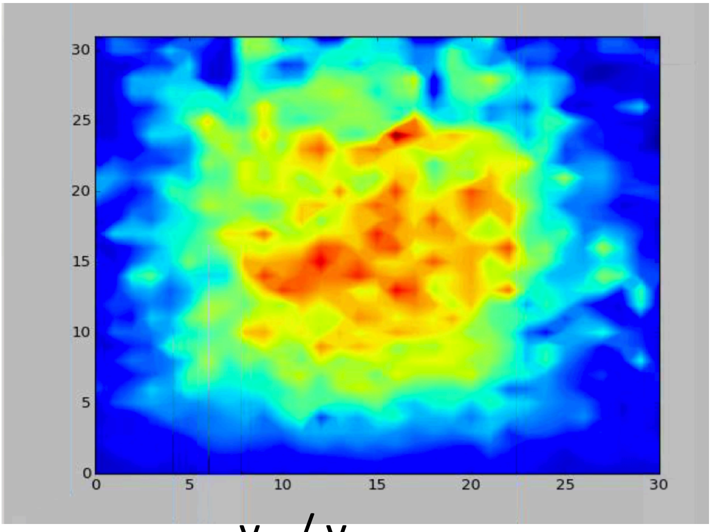
- Machine learning algorithm syntax simple, similar to scikit-learn, but Spark allows scaling

```
import adios as ad
import numpy as np
from pyspark.mllib.clustering import
BisectingKMeans

def read(ind):
    f = ad.file('/path/to/file')
    data = f['data_name'][:,ind[0]:ind[-1]+1,:]
    f.close()
    return data

def split(data):
    for d in np.rollaxis(data,1):
        yield d

Nnodes = 10
NcoresPerNode = 22
Nparts = Nnodes*NcoresPerNode*4
indices = np.array_split(np.arange(0,Nrecords),Nparts)

rdd1 = sc.parallelize(indices,Nparts)
rdd2 = rdd1.map(lambda v: read(v))
rdd3 = rdd2.flatMap(lambda v: split(v))

model = BisectingKMeans.train(rdd3, k=6)
```

Coherent phase space structures (blobs, holes, clumps, etc.)

- Various opinions on importance/long-term existence of phase space structures in strong turbulence [Dupree *Phys Fluids* 1972, Krommes PoP 1997, Kosuga NF 2017]

$v_\perp/v_{th}$



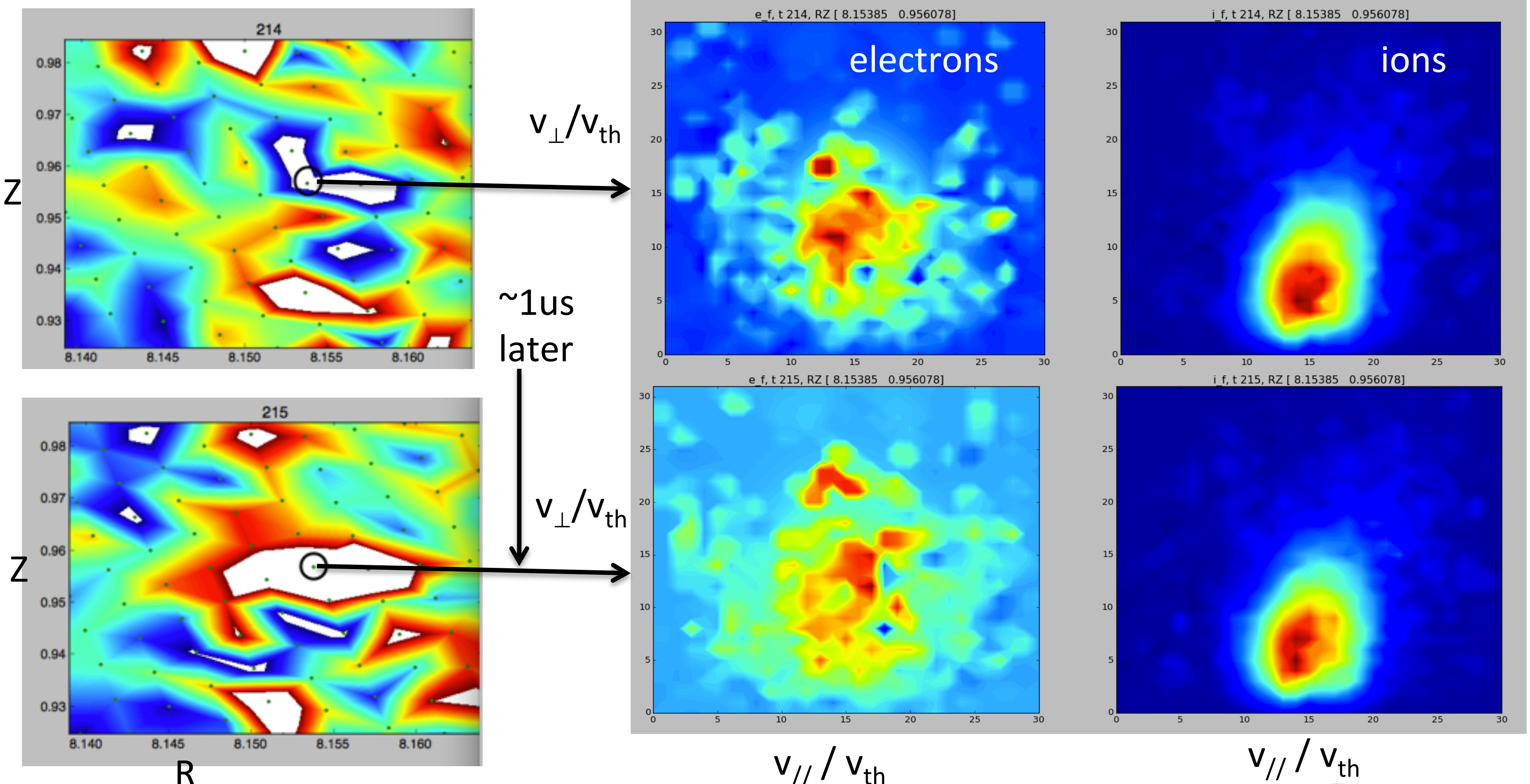
$v_{//} / v_{th}$

- Investigating single PDF from simulation can be misleading due to noise
- Apply K-means clustering to determine regions in velocity space which correlate well

Spark Motivation - can we find common signatures in velocity space?

Density fluctuations

Distribution functions

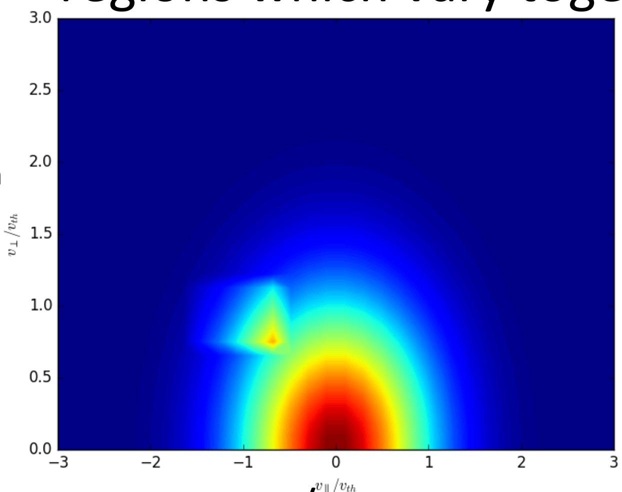


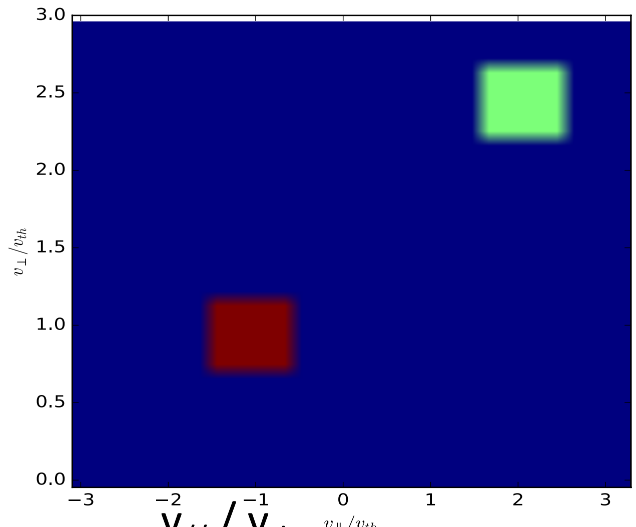
Synthetic data created to test k-means clustering with plasma distribution functions

- Maxwellian distribution function, with two square regions of velocity space with sinusoidal modulation:

$$\begin{cases} \cos(2\pi x) & -1.4 < v_{\parallel}/v_{th} < -0.5, \quad 0.75 < v_{\perp}/v_{th} < 1.22, \\ \cos(5\pi x) & 1.6 < v_{\parallel}/v_{th} < 2.6, \quad 2.25 < v_{\perp}/v_{th} < 2.72 \end{cases}$$

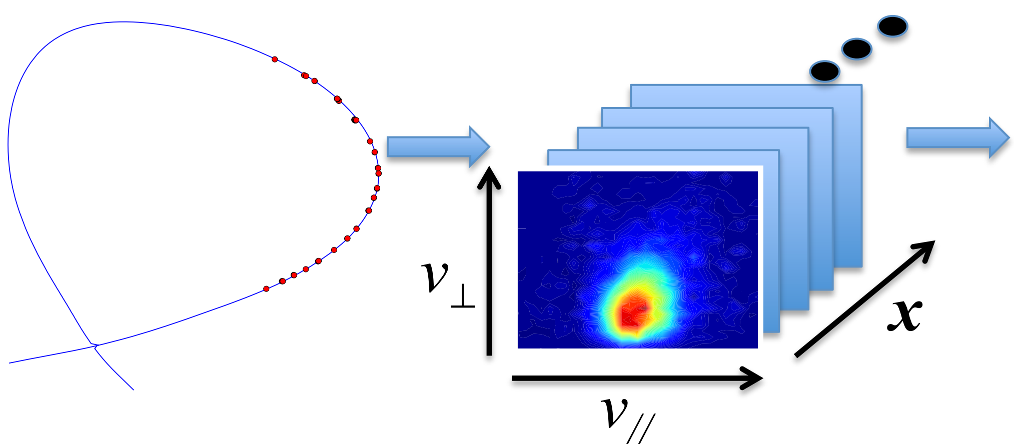
- K-means clustering with k=3 correctly separates the velocity space regions which vary together



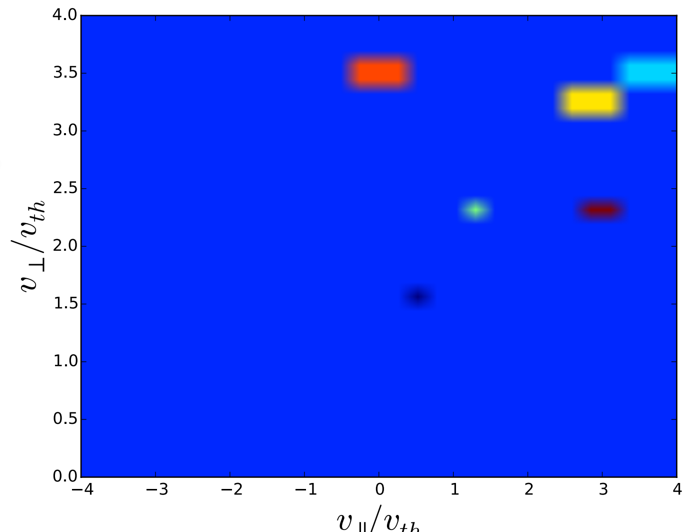


Bisecting K-means finds no direct structure in full edge region

- XGC1 distribution function set from ITER simulation, 500 GB/time step (only subset from single time-slice used, covering full pedestal edge region, 32 x 31 x ~8M = ~60GB)
- Bisecting K-means algorithm avoids issue of cluster initialization leading to local minima [Steinbach, 2000]
- Returned clusters noise based, subsequent runs change clusters found



[R.M. Churchill, *IEEE Proc. of NY Sci. Data Sum.*, 2016]

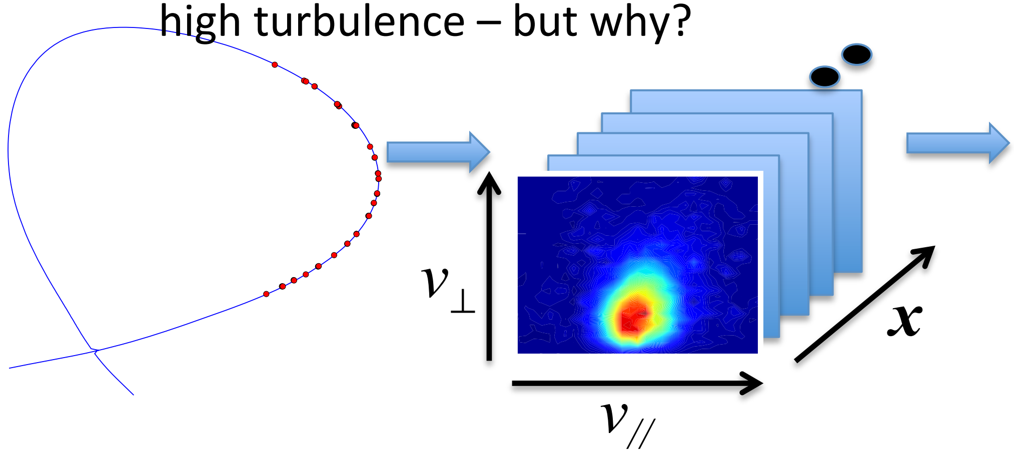


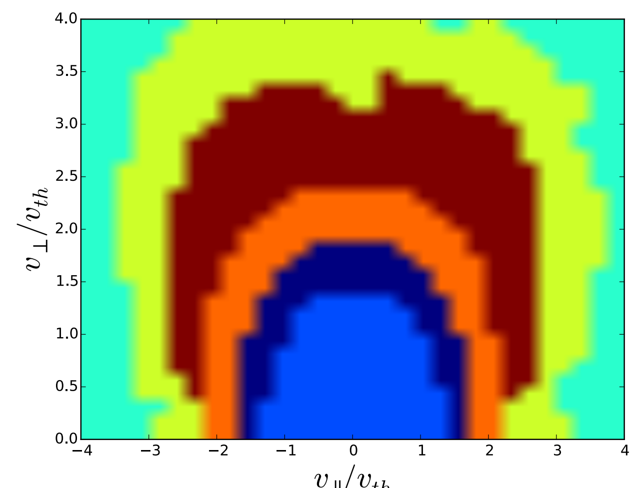
Bisecting K-means finds ring-like structure in turbulent spatial regions

- XGC1 distribution function set from ITER simulation, 500 GB/time step (only subset from single time-slice used,
covering only high turbulence regions in pedestal/SOL,
32 x 31 x ~60k = ~450MB)

- Bisecting K-means algorithm avoids issue of cluster initialization leading to local minima [Steinbach, 2000]

- Electron distribution function shows ring-like structure in spatial regions of high turbulence – but why?



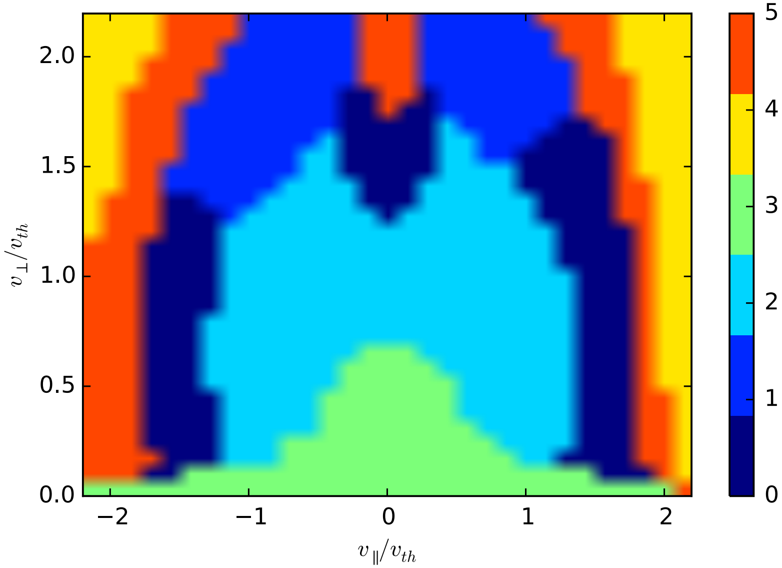


[R.M. Churchill, *IEEE Proc. of NY Sci. Data Sum.*, 2016]

K-means clustering after matching velocity space grid reveals more variable structure

- Renormalize all v-space grids onto same normalized grid

- Rerunning K-means clustering reveals more intricate structure
    - High energies ($E>E_{th}$) show break near trapped/passing boundary



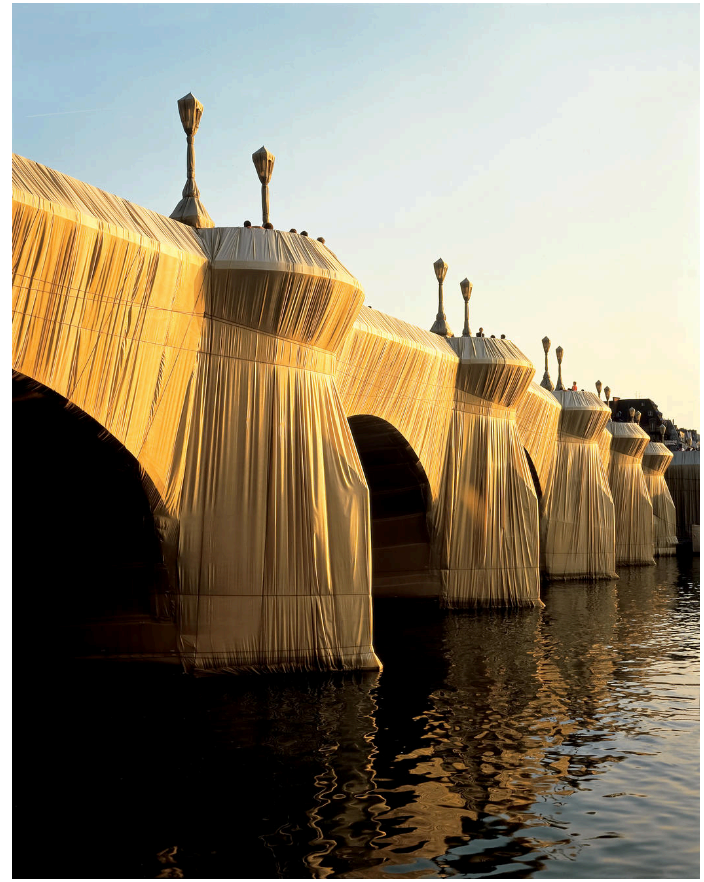
Summary

- Unsupervised machine learning can be used to search for structure in large data sets
- Apache Spark provides a simplified framework for distributed computing, including machine learning libraries
- K-means clustering on electron distribution functions from the gyrokinetic code XGC1 shows distinct structure in highly turbulent regions
    - Partial ring-like structure
    - separated at higher energies near the trapped/passing boundary

Background Slides

Christo and Jeanne-Claude

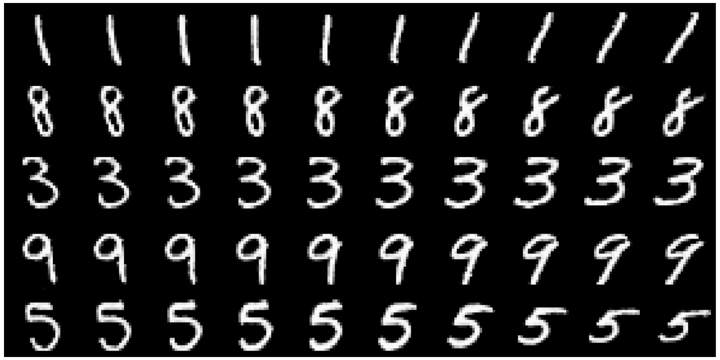
"While the intricate details of the structures are hidden, the essence of the structures are revealed all the while making the imposing and solid structure seem airy and nomadic"

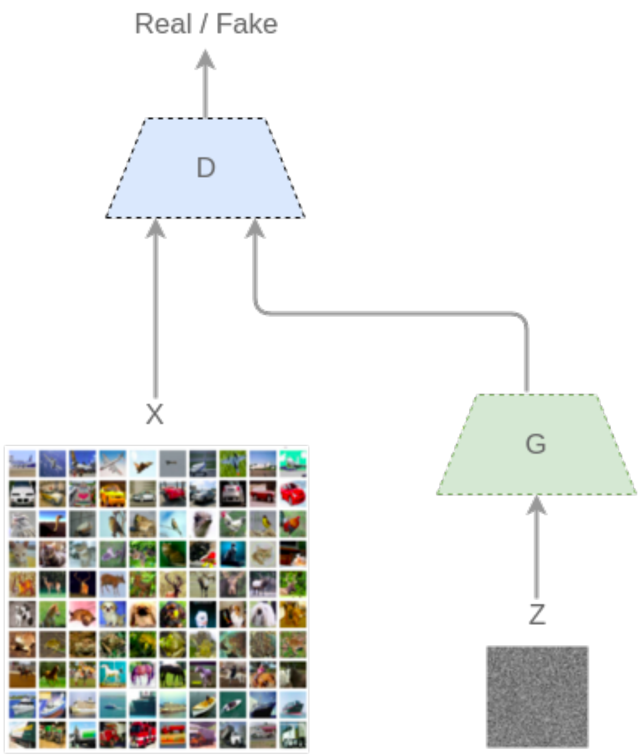


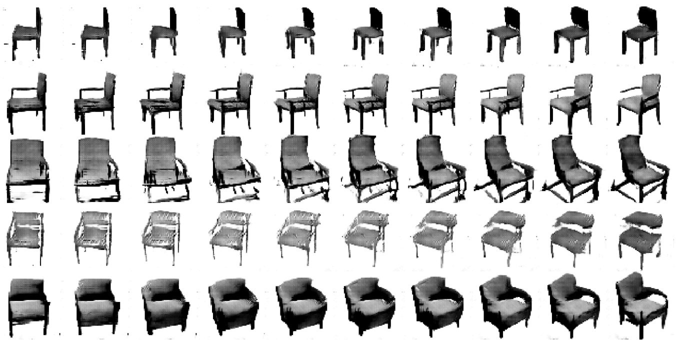
Future directions

Generative Adversarial Networks (GANs)

- InfoGAN: Maximizes mutual information for latent variables, allows for disentangled representation
  [Chen, NIPS 2016]

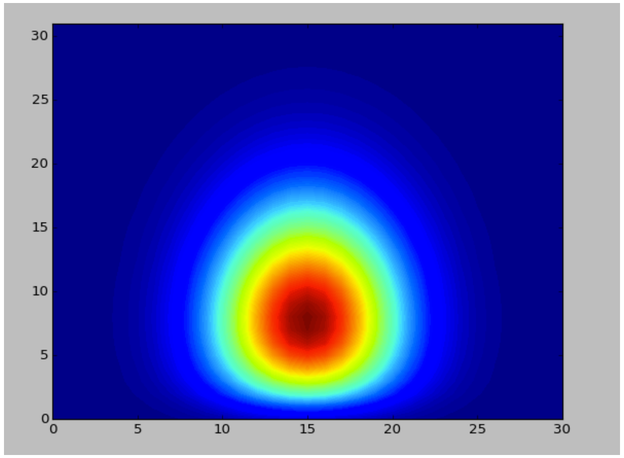






XGC1 core *f* distribution functions show little velocity space variation

f distribution functions from random core vertices were analyzed with K-means clustering



As expected, little variation was found